Graph-Convolved Factorization Machines for Personalized Recommendation

Yongsen Zheng, Pengxu Wei, Ziliang Chen, Yang Cao, and Liang Lin

Abstract—Factorization machines (FMs) and their neural network variants (neural FMs) for modeling second-order feature interactions are effective in building modern recommendation systems. However, feature interactions are based upon pairs of features, whereas multi-features correlations commonly arise in real-world financial product recommendation scenarios. We propose an effective neural recommender system, graph-convolved factorization machine (GCFM), with the spirit of the symbolic graph reasoning principle that provides lightweight and interpretable recommendation suggestions. Given a sample for the recommendation, GCFM constructs the corresponding dense feature embeddings and computes the sample-specific feature relationship graph. Then, a multi-filter graph-convolved feature crossing (GCFC) layer for feature embeddings establishes cross features with their neighboring embeddings. GCFM thus extends the feature interactions from pairs to neighbors to capture more comprehensive and explainable information while simultaneously reaping the advantages of representation learning. To exploit these capabilities, we apply a Graph Bayesian Optimization (GBO). During training, our GBO automatically optimizes our GCFM, including training hyperparameters and architecture hyperparameters. Besides, we conduct extensive experiments on two public financial applications benchmarks, USCFC and OTC, and two real-world datasets that we collect offline. Our GCFM significantly outperforms state-of-the-art algorithms and shows its interpretability in recommendation tasks. We further extend our model to online real-world applications, showing an appealing human-level decision intelligence in real scenarios.

Index Terms—Factorization Machines, Finance Product Recommendation System, Graph-Convolved Factorization Machines, Multi-Feature Interaction.

1 Introduction

Recommender Systems (RSs) [1] facilitate users to navigate large collections of items in a personalized way, which play an important role in a variety of domains, such as the electronic commerce [2], social network [3] and personalized advertising [4]. Nevertheless, complex, uncontrollable and changing user demographics and item properties invite severe feature interactions [5], which significantly degrade the recommendation performance. Learning sophisticated feature interactions is an essential component for RSs.

Recently, many research efforts have been devoted to k-order feature interactions [5], [6], [7], [8], [9]. This strategy groups exhaustively features with predefined numbers (*e.g.*, pair or 3-order), essentially employs full permutation among features and builds connections between each feature and others. A remarkable RS approach, Factorization Machine (FM) [5], extracts the second-order information by estimating the weights of pairwise features and thus learning a pairwise embedding vector. More recently, some neural variants of FMs, *e.g.*, DeepFM [6], NFM [10], AFM [7], xDeepFM [11], and FwFMs [12], have empowered second-order interaction mechanisms with deep networks in RSs. Furthermore, Higher-Order FMs (HOFMs) [8] also adopt a similar manner and just extend pairwise FMs by connecting features in groups with feature permutations.

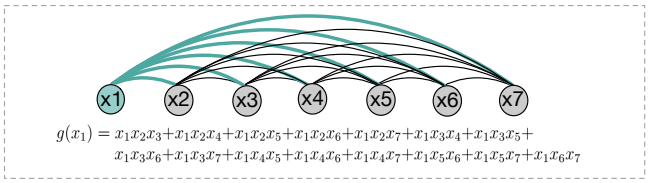
However, these FMs follow a rigid feature-connected strategy by arranging features into groups with k elements and exhaustively modeling all the possible feature-interactions for each group. Thus it fails to acquire domain-adaptive embeddings, which greatly limits their applications in practice. Meanwhile, this strategy is exhaustive since it considers all the feature-interactions no matter how closely related or negatively irrelevant they are. Therefore, these FM methods multiply these feature-interactions in sequence and are prone to feature vanishing derived from small feature-interaction weights.

Additionally, due to k-order feature interactions in a permutation manner, these methods involve extremely high-cost computations and thus in practice, existing RS methods consider 3-order feature interaction at most [5].

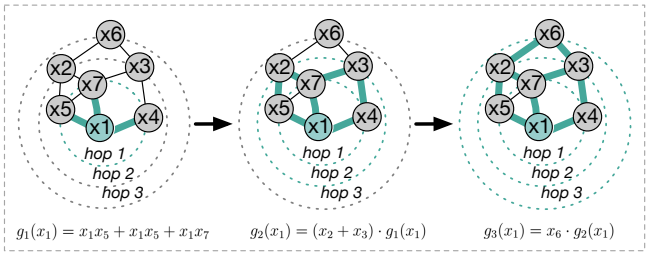
To address these problems, we propose a generalized FM model, Graph-Convolved Factorization Machine (GCFM), which flexibly and adaptively models multi-

- *Y. Zheng is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China, and is also with the Research Institute of SenseTime, Shenzhen, China. Email: z.yongsensmile@gmail.com*
- *P. Wei is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. Email: weipx3@mail.sysu.edu.cn*
- *Z. Chen is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. Email: zlchilam@163.com*
- *Y. Cao is with the Research Institute of SenseTime, Shenzhen, China. Email: caoyang@sensetime.com*
- *L. Lin is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China, and is also with the AI Research Institute of DarkMatter, Guangzhou, China. Email: linliang@ieee.org*

Pengxu Wei is the corresponding author.



(a) Traditional feature interactions



(b) Multi-feature interactions

Fig. 1. Illustration of k-order feature interaction widely explored and our multi-feature interaction with graph. The former follows a rigid feature-connected strategy by arranging features into groups with k elements and exhaustively modeling all the possible feature-interactions for each group. Obviously, it just considers predefined k-order feature interactions no matter how closely related or negatively irrelevant they are; due to a permutation manner, it invites extremely high-cost computations and thus in practice, existing RS methods consider 3-order feature interactions at most. However, our GCFM builds multi-feature interactions with graph adaptively to explore positive interactions among features and reduce negative impacts from non-connected features, which greatly reduces the computation cost. For brevity, we take x_1 as an example and the same operation for other all features. $g(x_1)$ in (a) and $g_i(x_1)$ in (b) represent the $(i+1)$-order feature interaction.

feature interactions in a graph.

Features from each sample are leveraged to compute the corresponding multi-feature interaction graph. This graph correlates each feature with their neighboring graph nodes. Accordingly, we develop a Graph-Convolved Feature Crossing (GCFC) layer in which the feature embedding is built on multi-feature interactions. This process traverses all features for each input example and propagates its influence on other features. After passing through the GCFC layer, the derived embeddings are compressed by a graph pooling layer and then concatenated together to feed a predictor head in GCFM, which produces our recommendation outcome. Besides, our GCFM is quite flexible and low-cost; particularly, the connections among features indicate their positive effects of interactions and suppress trivial or negative effects without connections from large and messy datasets. Thus, GCFM relates multiple features and provides an interpretable recommendation strategy in a lightweight architecture.

Considering intractable optimization settings, we further employ a Graph Bayesian Optimization method (GBO) to determine the hyperparameters of graph filters. This GBO evolves together with our GCFM training and thus is capable of automatically tune the learning and dropout rates for learning GCFM on the fly as well as the structure hyperparameters. Overall, it is a lightweight, flexibly implemented and highly effective learning strategy for optimizing GCFM.

We have conducted experiments on six recommendation datasets: four public benchmarks, *i.e.*, Criteo, Avazu, the US Consumer Finance Complaints dataset (shortly, USCFC) and Adult while the other two financial datasets are from our practical real-world projects of SenseTime Co. Ltd.[1], *i.e.*, Over The Counter (shortly, OTC) and Trading in the Field (shortly, TF). Our GCFM achieves state-of-the-art performance in comparison with related approaches. It outperformed a strategy based on finance-area expertise by 6.6% in terms of accuracy, thus showing human-level decision intelligence in real scenarios. Besides, we have analyzed GCFM qualitatively to verify its appealing interpretability.

Briefly, our contributions in this paper are threefold:

- We propose a novel Graph-Convolved Factorization Machine (GCFM) for the efficient neural recommendation. GCFM follows a graph reasoning principle to construct feature relation graphs and extends k-order interactions to multi-feature interactions with graph to adaptively capture more comprehensive information. This simple configuration makes our model effective and interpretable in a lightweight architecture.
- We exploit the GBO strategy to train our GCFM for intractable hyperparameter optimization. GBO concurrently determines the number of graph filters, pooling filters and even the size of the pooling kernels. In some sense, this method can be treated as an optimal architecture search approach for calibrating feature interaction graph construction, which is beneficial for real-world applications.
- We conduct extensive experiments to evaluate GCFM on four finance recommendation datasets and extend it to real-world finance applications. GCFM achieves a state-of-the-art performance and appealing interpretability for RS. Particularly, in real-world projects, significant progress has been achieved in comparison with an up-to-date expertise system, thus verifying its human-level recommendation intelligence.

The remainder of the paper is organized as follows. Sec.2 reviews previous approaches related to RSs and Sec.3 elaborates our proposed framework from the recommender system perspective. Our experimental results on six datasets are reported in Sec.4. Sec.5 extends our model to real-world applications and provides our online performance results. Sec.6 concludes the paper.

2 Related Work

2.1 Recommendation Systems

RSs [14] are widely utilized in a variety of areas, including movies [15], music [16] and social tags [17]. They produce a list of recommendations and predict a rating

1. https://www.sensetime.com/

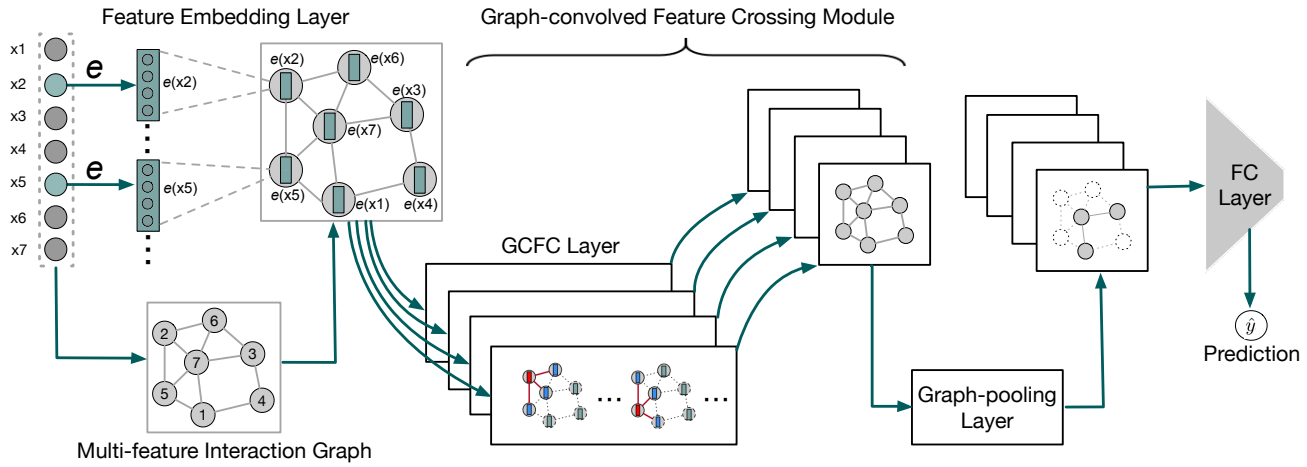


Fig. 2. Overview of our graph-convolved factorization machine (GCFM). At the very beginning, for each input $\mathbf{x} = (x_1, ..., x_n)$, its features are embedded in the form of low-dimensional feature vectors (embeddings). Simultaneously, $\mathbf{x}$ induces a mathematical representation of a graph over the features, namely, a binary adjacent matrix computed based on the Euclidean distances between the features. Then, our newly proposed graph-convolved feature crossing (GCFC) layer is used to construct feature interactions with graph. This operation involves graph-convolved filters acting on the feature embeddings with other embeddings in their neighborhoods. Subsequently, a nonlinear activation ReLU [13] is applied and then its output is passed to a graph-pooling layer. Finally, we leverage a fully connected layer as our predictor for financial product recommendation.

or preference that a user would assign to an item; thus, they are a subclass of information filtering systems [18]. RS approaches are typically categorized into two kinds, collaborative filtering [19] or content-based filtering [20].

Collaborative filtering approaches focus on collecting and analyzing a massive amount of information from user historical behaviors [21], implicit feedback [22], activity scope and predicting what users will have an interest in based on their similarity to other users. Collaborative filtering approaches need not necessarily rely on machine-analyzable content and therefore can offer accurate recommendations for complex items such as movies. Another common class of the RS approach is content-based filtering, in which a series of discrete characteristics of an item is considered to recommend additional items with similar attributes. In a content-based RS, keywords [23] play an important role in describing items, and a user profile [24] is built to indicate the types of items in which a particular user is interested.

Recently, deep learning has been revolutionizing recommendation architectures, achieving a significant RS performance. Deep learning [25] is capable of capturing nonlinear and nontrivial user-item relationships and enables the extraction of feature representations that capture intricate relationships in higher layers. Due to their ability to learn deep representations, deep learning architectures such as convolutional neural networks (CNNs) [26], recurrent neural networks (RNN) [27], deep semantic similarity Models (DSSMs) [28], restricted boltzmann machines (RBMs) [29], long short term memory (LSTM) [30] and generative adversarial networks (GANs) [31] have been widely applied in RSs [14].

2.2 Factorization Machines

FM is a widely-explored collaborative filtering based RS approach [19]. It aims to solve the problem of feature combination [32] in sparse feature vectors via one-hot encoding for enhancing performance on recommendations. Subsequently, a series of FM variants are proposed. FFM [9] aims to assign heterogeneous features to different fields to reduce noises for feature interactions. AFM [7] was developed with a focus on learning the importance of feature interactions via a neural attention network. Wide&Deep model [33] combines the benefits of memorization and generalization of a wide set of cross-product feature transformations and deep neural networks with less feature engineering. Deep&Cross network [34] keeps the benefits of a DNN model to make an automatic feature engineering and prevent exhaustive searching. However, these methods are limited to embedding pairwise feature interactions for RS and generally result in exhaustive efforts of computation. NFM [10] extends to learn high-order feature interactions that are seamlessly combined with the nonlinearity of deep neural networks. DeepFM [6] derives an end-to-end learning model that emphasizes both low and high order feature interactions. xDeepFM [11] incorporates a Compressed Interaction Network (CIN) and DNN for automatically learning high-order feature interactions in both explicit and implicit fashions. DIN [35] designs an activate related user behaviors and obtains an adaptive representation vector for user interests which varies over different ads. AutoInt [36] explores the multi-head self-attention mechanism to learn the high-order feature interactions. FiBiNET [37] aims to dynamically learn the feature importance and fine-grained feature interactions. FwFMs [12] can effectively capture the heterogeneity of field pair interactions by

introducing and learning a field pair weight matrix. PNN [38] refines feature interactions as field-aware feature interactions and extends FM with kernel product method.

2.3 Graph Convolution Recommendation Network

More recently, most researchers pay more attention on graph convolution recommendations. R-GCNs [39] introduces relational Graph Convolution Networks (GCNs) [40] and apply them to two standard knowledge-based completion tasks with the aid of knowledge graphs. [41] devises a Bayesian GCN to alleviate strong dependency on training data and prevents vulnerable decision making. [42] proposes an effective GCN based model for social recommendations, which mainly borrows the strengths of GCNs to capture user preferences. [43] develops a novel method based on highly efficient random walks to structure the convolutions and design a new training strategy to improve robustness and convergence of the model. [44] comprises a document encoder and a context encoder using GCNs to make accurate predictions without well-organized benchmarking datasets. [45] introduces an improved GCN model with high-order feature interaction to address the challenges of the lack of negative samples and a large number of candidate items. GC_MC [46] is a graph auto-encoder framework for the matrix completion task in recommender systems. GIN [47] introduces co-occurrence relationship of commodities to explore the potential preferences. Multi-GCCF [48] focuses on capturing the intrinsic difference between user-item and modeling user-user and item-item relationships explicitly. NGCF [49] proposes a novel neural network to model high-order connectivity in user-item graph. Fi-GNN [50] makes good use of the strong representative power of graphs to model sophisticated feature interactions.

The studies listed above mainly consider the quantity of data and ignore modeling multi-feature interactions. Among these methods, our work is the most relevance to [50]. Concretely, both [50] and our work model feature interactions via graph. However, [50] requires each node to interact with all remaining nodes in a full permutation manner, which is rigid and extremely high-cost and easily leads to feature vanishing. Our work builds multi-feature interactions with graph adaptively to explore positive interactions among features and reduce negative impacts from non-connected features.

3 METHODOLOGY

3.1 Overview

Existing methods exhaustively group features with certain numbers and consider their k-order feature interactions no matter how closely related or negatively irrelevant they are, which is extreme high-cost and easily leads to feature vanishing as discussed above. To address these problems, we proposed a novel Graph-Convolved Factorization Machine. GCFM constructs the multi-feature interaction graph to built connections among features and sequentially learns their trainable interaction weights in the graph-convoluted feature crossing module. Specially, we construct the multi-feature interaction graph from raw inputs and derive a binary adjacent matrix based on the Euclidean distance between features. Subsequently, GCFM learns adaptively graph filters, *i.e.*, interaction weights, and with them deploys the graph-convoluted feature crossing. During features crossing, each feature will interact with its neighbor features; in the process of crossing, the neighbor features propagate their feature messages to the central features respectively. Furthermore, a graph-convoluted feature crossing strategy avoids feature vanishing because it is able to strengthen closely related features and weaken negatively irrelevant features. Finally, the derived embeddings are compressed by a graph pooling layer and then are concatenated together to feed a predictor head in GCFM, which produces our recommendation outcome. The pipeline of our proposed GCFM is shown in Fig.2.

In the following, we will first revisit a conventional FM in Sec.3.2 and describe elaborately GCFM for recommendation decisions in Sec.3.3. In Sec.3.4, we provide details of the learning algorithm of GCFM, including model parameter optimization and a Graph Bayesian optimization for hyperparameters during training.

3.2 Factorization Machine

In the context of Recommender Systems, FMs have been cast into a supervised learning problem that aims ultimately to provide a suggestion based on collaborative recommendation [19], [51]. Compared with linear predictors [52], FMs additionally incorporate the second-order feature interactions by learning cross-feature weights.

Specifically, a sample of recommendation is presented as a n-dimensional real-valued vector $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{R}^n$, where x_j $(\forall j \in \{1, 2, ..., n\})$ denotes its j^{th} feature; $y \in \mathbb{R}$ represents its associated label indicating the preference of users. Accordingly, A 2-order FM predicts the label $\hat{y}$ of $\mathbf{x}$ as follows:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j, \quad (1)$$

where w_0 denotes the global recommendation bias; w_i $(i > 0)$ is the linear prediction weight of the i^{th} feature and $\langle \cdot, \cdot \rangle$ is the inner product of two vectors of size d, and $\langle \mathbf{v}_i, \mathbf{v}_j \rangle := \sum_{f=1}^{d} v_{i,f} \cdot v_{j,f}$. $\mathbf{v}_i$ is the i-th variable with d factors. $d \in \mathbb{N}_0^+$ is the hyperparameter that denotes the factorization dimensionality. Moreover, the 2-order FM can be generalized to a k-order FM:

$$\begin{aligned} \hat{y}(\mathbf{x}) :=& w_0 + \sum_{i=1}^{n} w_i x_i \\ &+ \sum_{l=2}^{k} \sum_{i_1=1}^{n} \cdots \sum_{i_l=i_{l-1}+1}^{n} \Big(\prod_{j=1}^{l} x_{i_j} \Big) \Big(\sum_{f=1}^{d_l} \prod v_{i_j,f}^{(l)} \Big) \end{aligned} \quad (2)$$

where the interaction parameters for the l-th interaction are factorized by the PARAFAC model [53] with the model parameters $\mathbf{V}^{(l)} \in \mathbb{R}^{n \times d_l}, d_l \in \mathbb{N}_0^+$ is the hyperparameter that defines the dimensionality of factorization.

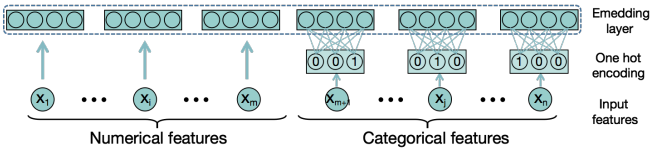


Fig. 3. Illustration of the embedding for an input with categorical and numerical features.

3.3 Graph-Convolved Factorization Machine

Instead of calculating k-order cross-feature terms similar to FMs, GCFM captures this information by considering the multi-feature interactions on a self-constructed graph. Benefiting from the graph representation, in the process of crossing, the neighbor features propagate their feature messages to the central feature for crossing, respectively.

3.3.1 Multi-feature interaction graph

We define a Multi-feature Interaction Graph (MIG) on all the training samples $\mathbf{X}$ as $\mathcal{G}(\mathbf{X}) = (\mathcal{V}, \xi, G(\mathbf{X}))(|\mathcal{V}| = h)$, where h is the number of feature nodes, $\mathcal{V}$ denotes the set of feature nodes and ξ denotes the set of their edges. The binary adjacency matrix $G(\mathbf{X}) \in \{0,1\}^{h \times h}$ indicates primary feature connection or interaction weights to measure the correlation among features. Specifically, $\mathbf{X}$ consists of categorical features (e.g., user ID, gender) and numerical features (e.g., age). As shown in Fig.3, each categorical feature is converted into a high-dimensional sparse features via field-aware one-hot encoding, which is applied widely in related works [6], [10] and [11]; each numerical feature is normalized between 0 and 1. Then, the numerical features and one-hot features are fed into an embedding layer to generate dense low-dimension feature vectors (*i.e.*, embeding vectors). We apply the Euclidean distance [54] to calculate the similarity between embedding vectors with the goal of obtaining $G(\mathbf{X})$, which are dynamically changing during training.

In brief, for each feature, we firstly calculate its distances from other features and pre-define a threshold β with the value of 0.5. If the distance is greater than the threshold, which indicates that there exists the close relationship between features, and we initialize the connection weight to 1; otherwise, it indicates that there is only a weak relationship between two features, and we initialize the connection weight to 0. That is, 1 means the connection weight is the trainable parameters while 0 means that the connection is eliminated. Generally, the threshold is between 0 and 1, the smaller value denotes the weaker relationship between features while the larger value means the stronger one. Moreover, the threshold is better set to greater than or equal to 0.5 for the sake of positive interactions facilitation and negative impacts reduction. In this work, on one hand, considering that choosing β as 0.5 achieves the best results on all the datasets, we set the threshold value β to be 0.5 for our GCFM model in the experiments on all the datasets. On the other hand, we also use our GCFM_GBO to adaptively search the optimal threshold value with our GBO technology. Accordingly, we build a MIG shown in Fig.4. The specific connection weights, namely feature interaction weights, are formulated as trainable graph filter parameters, which would be learned in the following multi-feature interaction module.

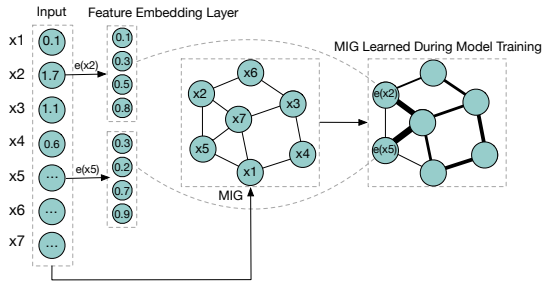


Fig. 4. Illustration of feature embedding layer and self-constructed MIG. The MIG is built by the raw input $\mathbf{x}$, and its node vector is the feature embedding of $\mathbf{x}$ while connection weights can be learned during the model training.

To represent each feature in embedding vectors, similar to FMs, a raw input $\mathbf{x}$ passes through a feature embedding layer parameterized by W_e, which transforms each feature of $\mathbf{x}$ into a low-dimensional embedding vector. Its embedding vector is described as follows:

$$e(\mathbf{x}) = [e(x_1), e(x_2), ..., e(x_n)] \in \mathbb{R}^{d \times n}, \tag{3}$$

as shown in Fig.3, where $e(x_i)$ is a d-dimensional embedding of the i^{th} feature and n is the number of feature fields.

Subsequently, the feature embedding vectors (*i.e.*, node vectors) and the interaction weights (*i.e.*, edge weights) will be learned during model training based on the basic graph MIG $G(\mathbf{X})$. For instance, as shown in Fig.4, the feature node x_7 has four neighborhood nodes including x_1, x_2, x_3, and x_5; thus x_7 will interact with these nodes with different edge weights which are trainable in the following graph-convolved feature crossing module.

3.3.2 Graph-convolved feature crossing module

Existing methods con k-order (*e.g.*, pair-wise or 3-order) feature-interactions no matter how closely related or negative irrelevant those features are. This manner easily leads to feature vanishing and exhaustive computation, which is greatly limited for practical applications. To address these issues, in our Graph-convolved feature crossing module, our GCFC layer is proposed to learn multi-feature interactions among multiple neighbors on our constructed MIG; for each feature as the anchor node, its neighbor nodes (features) propagate their node feature to it; after the anchor node receives the message, it interacts with these neighbor nodes with the learned interaction weights. A large weight can strengthen the positive effects, which means the closely related features, while non-connection weight means negatively irrelevant features.

Concretely, for the feature x_i, we take it as an anchor and choose its neighbors $\mathcal{N}(x_i)$ as its queries that interact with it on the MIG. Our GCFC projects the embedding

$e(x)$ into a graph embedding $w_g e(x)$ with a graph convolution filter w_g. Accordingly, we generates a cross-feature term $g(x_i)$ for the feature x_i as follows:

$$g(x_i) = \sum_{j \in \mathcal{N}(x_i)} [w_g e(x_i)]^T [w_g e(x_j)], \quad (4)$$

where $\mathcal{N}(x_i)$ represents the identities of the neighbors of the embeddings. Given M filters on $\mathcal{G}(\mathbf{X})$ to perform feature crossing, and connection weights W_G of $G(\mathbf{X})$ can be learned during the model training, thus we would have a derived convolution map with the m-th filter,

$$h^{(m)}(\mathbf{x}) = ([w_g^{(m)} e(\mathbf{x})]^T [w_g^{(m)} e(\mathbf{x})] \odot G(\mathbf{X})^T) \times \mathbf{1}^{n \times 1}. \quad (5)$$

After the feature crossing layer, a *ReLU* layer is deployed for capturing nonlinear information. These interactions are beneficial for exploring internal correlations among features.

Following the GCFC layer, we employ a graph pooling layer to cluster similar vertices in $\mathcal{G}(\mathbf{X})$ together and then to prune the redundant vertices, which reduces the computation and time cost for data pattern processing. Concretely, our graph-pooling layer first employs the Graclus multilevel clustering algorithm (GMC) [55] to transform each graph into a balanced binary tree. Then the nodes are ordered at the coarsest level (the graph with the fewest nodes), and this ordering is propagated to each finer level. At coarser levels, adjacent nodes are hierarchically merged such that the graph pooling operation becomes as efficient as 1D pooling [40]. Thus, we obtain the graph pooling outputs with the m-th filter $h^{(m)}(\mathbf{x}) = Graphpooling(h^{(m)}(\mathbf{x}))$.

In the context of recommendation, we mainly consider the binary classification with a given sample $\mathbf{x}$ with n features $\mathbf{x} \in \mathbb{R}^n$. Accordingly, we employ a fully connected layer parameterized by $\boldsymbol{\theta}$ to predict user preferences as follows:

$$\hat{y}(\mathbf{x}) = \boldsymbol{\theta}^T (h^{(1)}(\mathbf{x}), \cdots, h^{(M)}(\mathbf{x})), \quad (6)$$

where comma represents the operation of concatenation.

3.4 Learning

GCFM is parameterized by several hyperparameters for determining the network architecture and jointly improving the GCFM training efficiency, including *dropout rate* δ, *regularization parameter* λ, *the number of graph filters* M, *kernel size* μ and *the threshold* β *for feature connections on graph.* A large number of possible hyperparameter tuning combinations build an obstacle to maximizing the capability of GCFM in practice. Accordingly, in the learning phase of GCFM, we propose graph Bayesian optimization strategy to search for hyper-parameters for diverse recommendation scenarios. Given the learned hyper-parameters, we perform the optimization for model parameters.

3.4.1 Graph Bayesian hyperparameter optimization

Traditional Bayesian Optimization (BO) aims to search hyperparameters automatically to optimize the objective function. Hyperparameters in our GCFM involve not only the network learning (*i.e.*, dropout rate δ, regularization parameter λ, the number of graph filters M, kernel size μ) but also the construction of the multi-feature interaction graph (MIG) (*i.e.*, the threshold β for feature connections on graph). The construction of the graph usually relies on a threshold to determine which two nodes are connected. This threshold can be regarded as a hyperparameter to search in a learning-based method, rather than the empirical setting. However, the optimization of this graph hyperparameter and those for the network learning is different, since this graph hyperparameter directly affects the structure of the graph which is the precondition for the whole model training with the network hyperparameters. BO is not responsible for this challenging optimization issue derived from the construction of the feature interaction graph and cannot be directly applied to graph neural networks. To mitigate this issue, in our work, the proposed Graph Bayesian Optimization (GBO) aims to optimize our model GCFM on multi-feature interaction graph. GBO adaptively searches the threshold of feature connections, which is used to determine the node connections and the structure of MIG. It also automatically optimize other hyperparameters on MIG.

These hyperparameters are associated with each other, meaning that if we change one of them, then the rest will be altered accordingly. This situation leads to a large number of possible hyperparameter tuning combinations, which represents an obstacle to maximizing the capability of GCFM in practice. Our GBO is an effective methodology for globally optimizing a black-box function f. In our context, f is specified by the performance of GCFM on a validation dataset. In this scenario, GBO is applied to search for the optimal hyperparameter settings to reduce the time to discover the most desirable RS. Concretely, we define a hyperparameter domain ϑ =(β, δ, μ, M, λ) in a five-dimensional space. Given this domain ϑ on graph, our Gaussian Process (GP) prior [56] is built on a mean function μ: $\vartheta \to \mathbb{R}$ and a Gaussian kernel κ: $\vartheta^2 \to \mathbb{R}$. Furthermore, given p observations $D_p = \{(\vartheta_i, \hat{y}_i)\}_{i=1}^p$, $\hat{y}_i = f(\vartheta_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. A GP posterior $p(f|D_p)$ is parameterized with mean value μ_p and Gaussian kernels κ_p. We adopt the symbols $\mu_p(\vartheta)$ and $\kappa_p(\vartheta, \vartheta')$ for the posterior process, where ϑ' means the central points of Gaussian kernel function. Then, μ_m and κ_m can be represented as

$$\mu_m(\vartheta) = k^T \widehat{K} Y, \quad \kappa_m(\vartheta, \vartheta') = \kappa(\vartheta, \vartheta') - k^T \widehat{K} k'. \quad (7)$$

This GP posterior is used to define an *acquisition function* φ_t: $\vartheta \to \mathcal{R}$, which measures the effect on evaluating f at any ϑ, where k=$\{\kappa(\vartheta, \vartheta_i)\}_{i=1}^p$, $\widehat{K}$=$(K + \sigma^2 \boldsymbol{I})^{-1}$, K=$\{\kappa(\vartheta_i, \vartheta_j)\}_{i=1,j=i+1}^p$, Y=$\{\hat{y}_i\}_{i=1}^p$, k=$\{\kappa(\vartheta, \vartheta_i)\}_{i=1}^p$, and $k' = \{\kappa(\vartheta', \vartheta_i)\}_{i=1}^p$.

To optimize the function f over our domain ϑ in our GCFM on graph, a set of points $\{\vartheta_i\}_{i=1}^{t-1}$ have been used to evaluate f at time t, yielding observations $\{\hat{y}_i\}_{i=1}^{t-1}$. Then we maximize the *acquisition function* ϑ_t = $\text{argmax}_{\vartheta \in \chi} \varphi_t(\vartheta)$ and obtain an evaluation of f at ϑ_t. In which, the *ac-*

quisition function aims to control the ratio of mean value and covariance. For example, if the training process of our model is very slowly, we can only run one group of hyperparameters, thus we should choose the one with larger mean value. However, if we can still run 100 times with computing power, we should choose the one with larger covariance to explore the optimal hyperparameters.

3.4.2 Model parameter optimization

Similar to FM, the model parameters of the GCFM are optimized as follows:

$$\min_{\boldsymbol{\theta},W_e,W_G,\{w_g^{(m)}\}_{m=1}^{M}} \frac{1}{|\mathcal{D}|}\sum_{(\mathbf{x},y)\in\mathcal{D}}^{|\mathcal{D}|} l(\hat{y}(\mathbf{x}),y) + \frac{\lambda}{2}\Big(||\boldsymbol{\theta}||^2 + ||W_e||^2 + ||W_G||^2 + \sum_{m=1}^{M}||w_g^{(m)}||^2\Big), \tag{8}$$

where $l(\cdot,\cdot)$ is binary cross entropy loss. Since the graph pooling layer is differentiable, we can apply backpropagation [57] to update all parameters in the GCFM. Besides, we employ the l_2 norm to prevent overfitting [58] during training. This regularization is simultaneously imposed on the GCFC layer and its predictor, which are controlled by their trade-off coefficient λ. When λ is small, the structural loss more strongly depends on the model capability; when λ is large, more attention is paid to the model complexity. In addition to the l_2 regularizer, we also employ the dropout technique [59] when training the GCFM.

Algorithm 1 Learning of our GCFM.

Input:
Labeled dataset $\mathcal{D}$;
Mini-batch size N;
Max iteration T.
Output:
GCFM hyperparameter vector ϑ =(δ, μ, M, λ);
GCFM parameters $\boldsymbol{\theta}$, W_e, W_G, $\{w_g^{(m)}\}_{m=1}^{M}$.

1: **For** t=1 to T **do**
2: Sample N examples $\mathcal{D}^{(t)}$ in $\mathcal{D}$;
3: Construct binary adjacency graphs $\{G(\mathbf{x}_j)\}_{j=1}^{|\mathcal{D}^{(t)}|}$ ($\forall \mathbf{x}_j \in \mathcal{D}^{(t)}$);
4: Compute feature embedding matrices $\{e(\mathbf{x}_j)\}_{j=1}^{N}$ by the feature embedding layer in Eq.3;
5: Compute M_t cross features $\{\{w_g^{(m)}(\mathbf{x}_j)\}_{m=1}^{M_t}\}_{j=1}^{N}$ by GCFC layer in Eq.5;
6: Obtain $\{\{h^{(m)}(\mathbf{x}_j)\}_{m=1}^{M_t}\}_{j=1}^{N}$ by the graph pooling layer;
7: Obtain $\{\hat{y}(\mathbf{x}_j)\}_{j=1}^{N}$ by the predictor layer in Eq.6;
8: Update $\boldsymbol{\theta}$, W_e, W_G, and $\{w_g^{(m)}\}_{m=1}^{M}$ by backpropagation;
9: Obtain p observations D_p;
10: Define the posterior process by $p(\hat{y}^{(t-1)}|D_p)$ by GP prior;
11: Use the posterior process to define an acquisition function φ_t;
12: Maximize acquisition function ϑ_t=argmax$_{\vartheta\in\chi}\varphi_t(\vartheta)$;
13: An evaluation indicator $\hat{y}^{(t-1)}$ at ϑ_t;
14: Model is updated as $\hat{y}^{(t)} = \hat{y}^{(t-1)}$;
15: **EndFor**

4 Experiments

In this section, we provide our experimental results on six datasets: four public benchmarks and two financial datasets that we collect in practical projects. Four public benchmarks are Criteo, Avazu, USCFC and Adult datasets and our two collected financial datasets are called OTC (Over the Counter) and TF (Trading in the Field).

4.1 Experimental Settings

Datasets. We conducted experiments on six datasets as followings.

- *C*riteo[2] is a famous industry benchmark dataset for CTR prediction, which contains 45 million users' click records in 39 anonymous feature fields on displayed ads. The goal is to predict the probability that the user will click on a given ad from his/her visiting page.

- *A*vazu[3] is widely used in many CTR model evaluations. It contains users' click logs with 40 millions of data instances. There are 23 feature fields indicating elements of a single ad impression.

- *U*SCFC[4] is available from the Consumer Financial Protection Bureau (CFPB), which sends thousands of consumer complaints about financial products and services to companies for responses and then publishes those complaints. 336,403 records of consumer complaints are sampled from different contexts, including attributes such as product, sub-product, issue, company, and ZIP code. A target value of 1 means the consumer has complained about financial products.

- *Adult*[5] has been widely applied in prediction tasks to determine whether a person makes over $50k per year. This dataset contains 48842 instances with 14 attributes, including categorical and integer variables. A target value of 1 means a person makes over $50k per year.

- *O*TC comes from our real-world project for actual trading records on famous investment securities. We sampled 89824 real fund trading logs drawn from the transaction data of one year and these fund trading logs were evenly distributed over 12 months. Then we randomly paired two negative samples with each log. A target value of 1 means a consumer has purchased the OTC financial products.

- *T*F is also from our real-world project for actual trading records on famous investment securities. We sampled 50000 real fund trading logs drawn from the transaction data of one year and these logs covered 12 months equally. Then we randomly paired two negative samples with each

2. https://www.kaggle.com/c/criteo-display-ad-challenge
3. https://www.kaggle.com/c/avazu-ctr-prediction
4. https://www.kaggle.com/cfpb/us-consumer-finance-complaints
5. http://archive.ics.uci.edu/ml/datasets/Adult

TABLE 1
Effectiveness Comparison of Different Algorithms. We highlight that our proposed model almost outperforms all baselines on six datasets. Further analysis is provided in Section 4.3.

Method	Criteo		Avazu		USCFC		Adult		OTC		TF	
	AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss
FM [5]	0.7836	0.4700	0.7706	0.3856	0.7768	0.4146	0.8117	0.3843	0.8250	0.3704	0.8441	0.3707
AFM [7]	0.7938	0.4584	0.7718	0.3854	0.7792	0.4132	0.8121	0.3838	0.8332	0.3693	0.8478	0.3648
HOFM [8]	0.8005	0.4508	0.7701	0.3854	0.7798	0.4128	0.8156	0.3817	0.8358	0.3672	0.8512	0.3612
Deep&Cross [34]	0.8012	0.4513	0.7643	0.3889	0.7796	0.4127	0.8153	0.3806	0.8364	0.3637	0.8539	0.3597
DeepFM [6]	0.8010	0.4514	0.7651	0.3881	0.7812	0.4112	0.8172	0.3789	0.8367	0.3645	0.8543	0.3594
xDeepFM [11]	0.8091	0.4461	0.7808	0.3818	0.7856	0.4103	0.8223	0.3765	0.8409	0.3635	0.8598	0.3521
AutoInt [36]	0.8061	0.4454	0.7752	0.3823	0.7865	0.4089	0.8259	0.3743	0.8415	0.3634	0.8612	0.3516
FiBiNet [37]	0.8021	0.4495	0.7803	0.3800	0.7873	0.4078	0.8262	0.3727	0.8411	0.3621	0.8612	0.3519
Fi-GNN [50]	0.8062	0.4453	0.7762	0.3825	0.7889	0.4054	0.8278	0.3701	0.8418	0.3612	0.8615	0.3494
GCFM	**0.8082**	**0.4425**	**0.7811**	**0.3792**	**0.7943**	**0.4021**	**0.8312**	**0.3664**	**0.8498**	**0.3587**	**0.8645**	**0.3467**
GCFM_GBO	**0.8088**	**0.4415**	**0.7815**	**0.3783**	**0.7966**	**0.4002**	**0.8323**	**0.3643**	**0.8515**	**0.3567**	**0.8657**	**0.3459**

log. A target value of 1 means a consumer has purchased the TF financial products.

Following the same experimental settings of [7], each dataset was split into three parts: 70% for training, 20% for validation and 10% for testing. We adopted two popular metrics Area Under the ROC (AUC) and Logloss to evaluate the performance of all methods. A higher AUC indicates a better performance while a lower Logloss denotes a better performance. It is worth noting that a slightly higher AUC or lower Logloss at ***0.001-level*** is regarded significant for CTR prediction task [6], [33], [34].

Implementation details: All the experiments are performed on the TensorFlow platform [60], which is an open-source library for machine learning and machine intelligence. Moreover, our GCFM mainly consists of an embedding layer, two graph-convolved feature crossing layers, a graph pooling layer and a fully connected layer. The numbers of input features are 18 for the OTC dataset, 12 for the TF, 14 for Adult, 13 for USCFC, 39 for Criteo, and 23 for Avazu. The embedding dimension is 8 for OTC, TF, USCFC, Adult datasets while 12 for Criteo and 10 for Avazu.

4.2 Comparison results with state-of-the-arts

To demonstrate the effectiveness of our method, we compare our proposed model with the following state-of-the-art methods. In particular, by comparing our proposed GCFM with each state-of-the-art method with different hyperparameters (such as the dropout rate and regularization parameter), we analyzed each component of our proposed method to clarify their contributions to its performance.

- *FM* [5] is a classic factorization machine approach and models all interactions between features using factorized parameters.

- *HOFM* [8] is a higher-order FM approach and extends FM to higher-order feature combinations.

- *AFM* [7] is an attentional FM approach and focuses on learning the weights of feature interactions via attention networks.

- *Deep&Cross* [34] has a architecture with a stack of 5 residual units. Deep&Cross leverages a deep learning technique to explicitly model feature interactions. We implemented this method based on the descriptions given in the paper.

- *DeepFM* [6] is an FM based neural network, which emphasizes both low- and high-order feature interactions by training a deep component and an FM component jointly.

- *xDeepFM* [11] incorporates CIN and DNN in an end-to-end framework, which can automatically learn high-order feature interactions in both explicit and implicit fashions.

- *AutoInt* [36] is a novel CTR prediction model based on self-attention mechanism, which is capable of automatically learning high-order feature interactions in an explicit fashion.

- *FiBiNet* [37] is a new model for dynamically learning the feature importance and fine-grained feature interactions.

- *Fi-GNN* [50] is a novel algorithm considering the multi-field features as a structured combination of feature fields, where each node corresponds to a feature field and different fields can interact through edges.

We compare our GCFM method with state-of-the-art methods on all six datasets, namely, Criteo, Avazu, USCFC, Adult, OTC, and TF. Table 1 summarizes our comparison of experimental results with state-of-the-art methods. It is observed that GCFM achieves the best performance overall baseline methods on six datasets. This is because our proposed GCFM builds multi-feature interactions with graph adaptively to explore positive interactions among features and reduce negative impacts from non-connected features. In this way, GCFM can alle-

TABLE 2
p for two-tailed t-test in terms of ***AUC*** on six datasets. If p is less than 0.05, it indicates that two models are significantly different.

Criteo Dataset											
Method	FM	AFM	HOFM	DC	DeepFM	xDeepFM	AutoInt	FiBiNet	Fi-GNN	GCFM	GCFM_GBO
GCFM	5.64e-14	2.30e-14	3.33e-13	3.11e-14	2.83e-13	0.002	3.54e-09	9.29e-14	0.004	/	5.23e-06
GCFM_GBO	5.20e-14	2.49e-14	3.68e-14	6.75e-14	3.48e-13	0.0001	1.11e-09	1.82e-13	0.0006	5.23e-06	/
Avazu Dataset											
Method	FM	AFM	HOFM	DC	DeepFM	xDeepFM	AutoInt	FiBiNet	Fi-GNN	GCFM	GCFM_GBO
GCFM	2.72e-14	2.10e-13	2.06e-13	8.29e-16	5.20e-16	0.01	5.10e-12	5.75e-05	1.92e-11	/	0.001
GCFM_GBO	3.47e-14	2.26e-13	2.08e-13	1.15e-15	8.03e-16	3.50e-05	4.73e-12	3.34e-06	1.61e-11	0.001	/
USCFC Dataset											
Method	FM	AFM	HOFM	DC	DeepFM	xDeepFM	AutoInt	FiBiNet	Fi-GNN	GCFM	GCFM_GBO
GCFM	2.66e-15	1.93e-15	3.43e-14	2.52e-15	5.35e-15	1.66e-13	3.01e-13	9.41e-13	8.20e-11	/	1.04e-08
GCFM_GBO	1.71e-15	1.33e-15	1.66e-14	1.68e-15	3.23e-15	5.61e-14	8.86e-14	2.17e-13	8.07e-12	1.04e-08	/
Adult Dataset											
Method	FM	AFM	HOFM	DC	DeepFM	xDeepFM	AutoInt	FiBiNet	Fi-GNN	GCFM	GCFM_GBO
GCFM	1.30e-15	4.95e-15	6.03e-15	1.11e-14	1.88e-14	5.98e-13	4.52e-11	5.22e-11	2.00e-08	/	7.83e-06
GCFM_GBO	4.40e-18	1.25e-16	8.22e-18	1.00e-16	5.48e-17	5.72e-16	1.28e-13	6.66e-14	3.95e-10	7.83e-06	/
OTC dataset											
Method	FM	AFM	HOFM	DC	DeepFM	xDeepFM	AutoInt	FiBiNet	Fi-GNN	GCFM	GCFM_GBO
GCFM	1.01e-17	9.30e-16	4.55e-15	1.10e-13	6.50e-15	1.44e-13	4.77e-13	4.25e-14	1.50e-12	/	2.68e-07
GCFM_GBO	1.32e-16	4.75e-15	1.83e-14	1.50e-13	2.67e-14	3.73e-13	8.33e-13	2.13e-13	1.81e-12	2.68e-07	/
TF Dataset											
Method	FM	AFM	HOFM	DC	DeepFM	xDeepFM	AutoInt	FiBiNet	Fi-GNN	GCFM	GCFM_GBO
GCFM	2.12e-16	3.25e-15	3.34e-14	1.74e-14	1.58e-13	1.01e-11	1.73e-09	9.83e-10	1.73e-09	/	1.55e-06
GCFM_GBO	4.10e-16	4.40e-15	3.51e-14	2.74e-14	1.46e-13	5.65e-12	2.63e-10	1.74e-10	2.63e-10	1.55e-06	/

TABLE 3
p for two-tailed t-test in terms of ***Logloss*** on six datasets. If p is less than 0.05, it indicates that two models are significantly different.

Criteo Dataset											
Method	FM	AFM	HOFM	DC	DeepFM	xDeepFM	AutoInt	FiBiNet	Fi-GNN	GCFM	GCFM_GBO
GCFM	2.30e-14	3.52e-14	9.05e-13	9.24e-12	3.57e-12	2.26e-09	2.59e-08	1.33e-11	2.06e-08	/	0.001
GCFM_GBO	3.08e-14	8.68e-14	2.79e-12	1.39e-11	6.68e-12	2.21e-09	1.57e-08	2.54e-11	1.27e-08	0.001	/
Avazu Dataset											
Method	FM	AFM	HOFM	DC	DeepFM	xDeepFM	AutoInt	FiBiNet	Fi-GNN	GCFM	GCFM_GBO
GCFM	1.74e-11	4.08e-11	3.01e-10	3.42e-12	1.49e-11	8.04e-08	1.33e-08	1.28e-05	9.68e-09	/	2.60e-05
GCFM_GBO	3.26e-12	8.79e-12	7.20e-11	1.08e-12	5.12e-12	3.12e-09	1.05e-09	6.06e-08	7.76-10	2.60e-05	/
USCFC Dataset											
Method	FM	AFM	HOFM	DC	DeepFM	xDeepFM	AutoInt	FiBiNet	Fi-GNN	GCFM	GCFM_GBO
GCFM	1.91e-13	1.57e-12	3.16e-12	3.06e-11	2.42e-11	2.04e-11	5.54e-11	2.96e-10	4.03e-08	/	4.52e-06
GCFM_GBO	1.40e-13	8.61e-13	1.57e-12	1.21e-11	9.29e-12	7.63e-12	1.66e-11	6.06e-08	2.66e-09	4.52e-06	/
Adult Dataset											
Method	FM	AFM	HOFM	DC	DeepFM	xDeepFM	AutoInt	FiBiNet	Fi-GNN	GCFM	GCFM_GBO
GCFM	2.18e-14	2.33e-14	2.74e-15	8.83e-13	2.34e-13	1.45e-12	1.38e-11	1.28e-05	5.18e-09	/	6.04e-07
GCFM_GBO	3.55e-14	3.99e-14	1.26e-14	7.72e-13	3.18e-13	1.51e-12	9.29e-12	2.25e-10	7.92e-10	6.04e-07	/
OTC Dataset											
Method	FM	AFM	HOFM	DC	DeepFM	xDeepFM	AutoInt	FiBiNet	Fi-GNN	GCFM	GCFM_GBO
GCFM	1.40e-10	2.80e-12	5.98e-13	5.53e-10	4.28e-10	1.39e-10	4.21e-10	4.88e-10	2.31e-08	/	5.32e-07
GCFM_GBO	2.06e-11	9.75e-13	1.97e-13	6.21e-11	6.36e-11	1.68e-11	4.70e-11	3.18e-11	7.44e-10	5.32e-07	/
TF Dataset											
Method	FM	AFM	HOFM	DC	DeepFM	xDeepFM	AutoInt	FiBiNet	Fi-GNN	GCFM	GCFM_GBO
GCFM	1.78e-15	3.56e-14	5.69e-13	6.18e-14	7.92e-14	8.04e-08	2.37e-10	5.54e-11	7.71e-08	/	6.37e-05
GCFM_GBO	1.63e-15	2.85e-14	4.00e-13	4.80e-14	6.04e-14	4.42e-10	7.82e-11	2.10e-11	9.68e-09	6.37e-05	/

viate the problem of feature vanishing derived from small feature-interaction weights. On Criteo dataset, GCFM and GCFM_GBO slightly underperform xDeepFM in terms of AUC, but xDeepFM is much worse than our methods. As for CTR prediction, it is required to use the predicted probability (measured by Logloss) to estimate the benefit of a ranking strategy (measured by AUC) for identifying positive instance and negative one [11]. Accordingly, our GCFM and GCFM_GBO perform better to predict CTR. Deep&Cross and DeepFM fail to guarantee improvement over prediction performance despite being able to capture high-order feature interactions. This may attribute to the fact that they learn feature interaction in an implicit fashion. AutoInt and FiBiNet perform better than FM-based methods, this is due to that they use attention or SENET mechanism to learn explicit feature importance. Comparing with the proposed GCFM, our GCFM_GBO achieves the better performance, this due to an automatic searching of hyperparameters, thereby indicating the effectiveness of GBO technique.

Generally, Our GCFM method is superior to all other methods for three main reasons: 1) the considered feature interactions involve multiple features on the graph; 2) each central node feature always intersects with its neighbors; 3) multi-feature interactions could strengthen their positive effects with large connection weights and weaken their negative effects with non-connected weights.

4.3 Significant test

In order to further verify the effectiveness of the proposed model GCFM and GCFM_GBO, we have provided two-tailed t-test between our model and all the compared models in the experiments. Concretely, 5-fold cross-validations on each dataset have been conducted, and then two-tailed t-test is conducted based on those results on each dataset. In the two-tailed t-test, p is the core index to measure whether there is a significant difference between two models [61]. If p is less than 0.05 (*i.e.*, significant level), it indicates that the significant difference exits between two models. The evaluation results of p for two-tailed t-test on six datasets in terms of two metrics are provided in Table 2 and Table 3, respectively. We observe that all the values of p between the proposed models (*i.e.*, GCFM and GCFM_GBO) and all the compared models in terms of two metrics (AUC and Logloss) on all the datasets are less than 0.05. This indicates that there is a

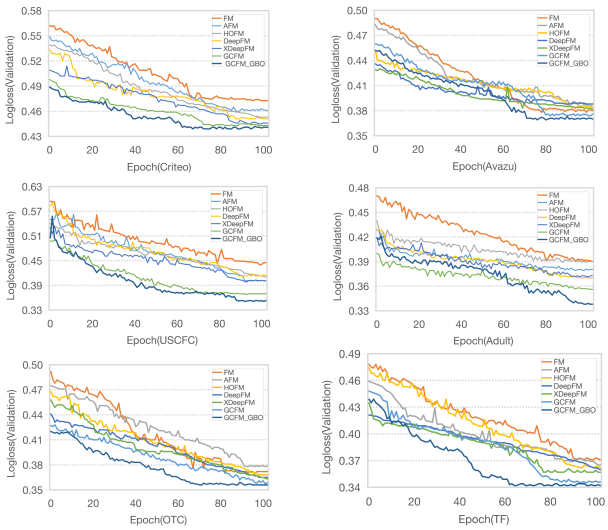


Fig. 5. Validation errors of GCFM and FMs-based methods w.r.t. different epochs. Best viewed in color.

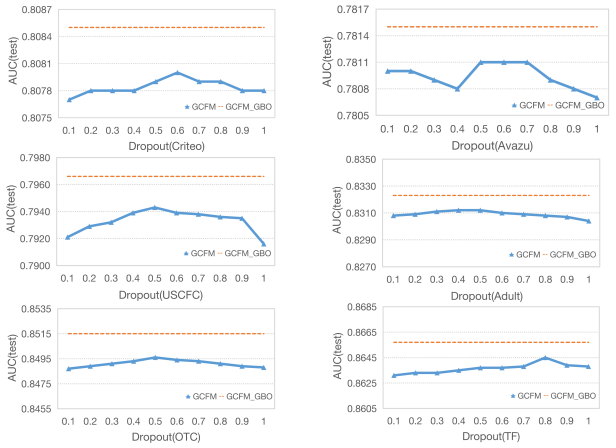


Fig. 6. Test errors of the GCFM and GCFM_GBO methods *w.r.t.* different dropout rates on the six datasets. Best viewed in color.

significant difference between the proposed model and all the baseline models. This further verifies the effectiveness of the proposed model.

4.4 Model Evaluation

In this section, we conduct a series of experiments about model evaluation, including ablation study on GBO, evaluation on hyperparameters and evaluation on the number of GCFC layers.

4.4.1 Evaluation on GBO

We conduct ablation study on GBO on six datasets in this section to verify the effectiveness of our GCFM_GBO. As shown in Table 1, GCFM_GBO outperforms the state-of-the-arts on six datasets. Besides, we conduct a group of comparison experiments about GCFM and GCFM_GBO from Fig.7 to Fig.8. For example, as shown in Fig.6, it is observed that GCFM_GBO outperforms GCFM in the setting of *dropout*, this because it can adaptively search the hyperparameters to optimize model. Especially, GCFM_GBO has quick convergence in Fig.5. The similar observation can be find in the setting of *lambda* and *learning rate*.

4.4.2 Evaluation on hyperparameter

Hyperparameters are generally intractable for stable training of deep network based models and would effect greatly the model performance. Thus, in this section, we conduct a comprehensive study on their influences on Criteo, Avazu, USCFC, Adult, OTC and TF datasets. Our involved hyperparameters are dropout rate, regularization parameter λ, and learning rate.

Dropout rate: The dropout rate [59] is the probability that a neuron will be kept in the network during training. Dropout is a regularization technique for reducing overfitting in neural networks and predicting unseen data more accurately. We set the dropout rate to 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0. As shown in Fig.6, all models achieve their own best performance when the dropout rate is properly set (between 0.6 and 0.8). The results show that reasonably increasing the randomness of a model can improve its generalization.

Regularization parameter λ: λ is a regularization parameter to limit the model complexity. This parameter can balance data fitting with rules for preventing overfitting. We set λ to 0, 0.005, 0.01, 0.02, 0.03, 0.04 and 0.05. As shown in Fig.7, on the USCFC dataset, all models achieve their own best performance when λ is set to 0.005. The results show that the smaller value of a λ can effectively prevent overfitting.

Learning rate: Learning rate is an important hyperparameter in supervised learning, which determines whether the objective function can converge to the local minimum and when it converges to the minimum. We set the learning rate to 1e-3, 1e-5, 1e-7, 1e-9, and 0.01. As shown in Fig.8, GCFM achieves the optimal performance when the learning rate is set to be a smaller value. The results show that a proper learning rate can make the proposed GCFM converge smoothly.

4.4.3 Evaluation on the number of GCFC layer

In this section, we evaluate the model depth (*i.e.*, the number of GCFC layers) fpr recommendation. We report results with 5-fold cross-validation on Criteo, Avazu, USCFC, Adult, OTC and TF datasets. On each cross-validation split, we train for 100 epochs using the Adam optimizer [62] with the learning rate 0.001. Other hyperparameters are set to be the same. Results are summarized in Fig.9.

We observe that GCFM achieves the highest performance with 2 or 3 layers on four datasets. For models with more than 3 layers, they maintain a stable performance.

4.4.4 Interpretability of recommendations

For most existing methods, feature interactions are usually constructed through permutation and combination operations. It is difficult to interpret the models due

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3100564, IEEE Transactions on Knowledge and Data Engineering

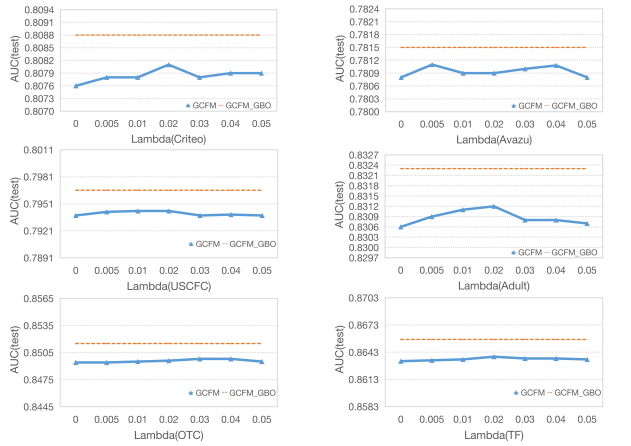


Fig. 7. Test errors of the GCFM and GCFM_GBO methods *w.r.t.* different λ values on the six datasets. Best viewed in color.

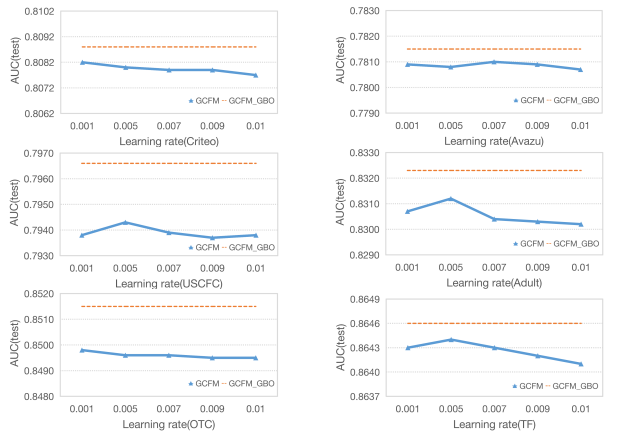


Fig. 8. Test errors of the GCFM and GCFM_GBO methods *w.r.t.* different learning rates on the six datasets. Best viewed in color.

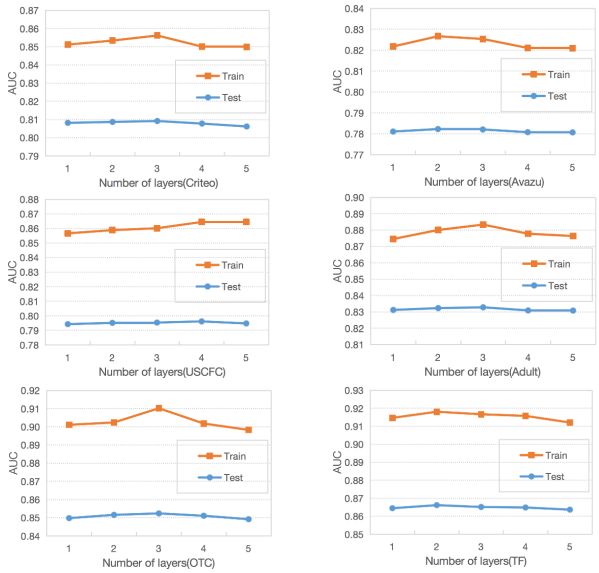


Fig. 9. Evaluation of the model depth (*i.e.*, the number of GCFC layers) on classification performance. Markers indicate mean classification accuracy (training vs. testing) for 5-fold cross-validation.

to the complex transformations involved. In the GCFM framework, this dilemma is resolved by the design of the GCFC layer. To explain its mechanism, we visualize all MIG of our four datasets shown in Fig.10, and the Fig.11 shows the similarity between an anchor features and other features with of FM, which also provide the convincing result for our interpretability.

For OTC, we can see that the feature *tra_name* has eight neighbor features, which are *zipcode*, *fund_int*, *cust_code*, *channel_mode*, *currency_diversity*, *date_id*, *profession_code*, and *fund_code*. Namely, the feature *tra_name* has close relationship with these features. Apparently, in our real-life, *fund_code*, *profession_code* and *cust_code* determine the *trd_name*. *Currency_diversity* and *fund_intl* also effect the *tra_name*. This manner is consistent with human behavior.

For TF, the feature *rise_info* impacts *tra_date*, which accords with actual fund trade. Meanwhile, the feature *tra_name* is relevant with *secu_code*, which acts like the OTC dataset. In fund trading, the fund code plays an important on the transaction name and this behavior gives a good explanation of fund trading.

For Adult, the feature *workclass* is affected by multiple features such as *age*, *race*, and *occupation*. Obviously, the *occupation* and *age* are the important factors for *workclass*. For example, if a person is 10 years old, his occupation may be a student and unlikely a scientist. This interpretability is consistent with our human understanding.

For USCFC, the feature *complaint_id* has seven neighbor features, which means this feature is related to these seven features, which play a significant role in *complaint_id*. For instance, the feature *timely_response* has an influence on the central feature *complaint_id* because the earlier the time, the smaller the *complaint_id*. It provides more reliable proof for our interpretability.

Overall, our GCFC layer provides interpretability of our model in two aspects. The first one is finding the neighborhood features for each feature in the graph, which surprising accords with human logical thinking because humans practically will narrow down the field when seeking some relationship between objects. If the neighborhood features for each feature cannot be found, then each feature must interact with the rest of the features, which will lead to unexplainable behavior and uncontrolled noise. The second one is weighting the different features. Normally, different features have different relationships with each other. The GCFM layer performs this behavior via a training model. The weights from model training are used to judge whether an association is strong enough to imply a meaningful causal relationship. Thus, our GCFM is consistent with human behavior, which facilitates the interpretability of our model.

5 Real-World Applications

In Section 4, we follow the prevalent evaluation metric similar to existing methods in RS research community and conduct extensive experiments on four public benchmarks

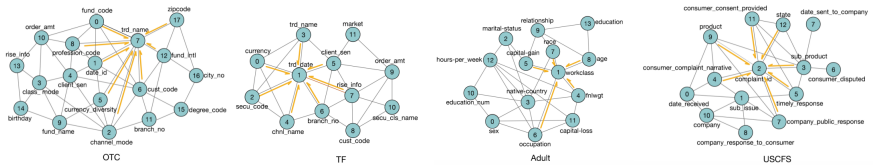


Fig. 10. Our MIGs in the proposed GCFM. On four datasets, we show the constructed MIGs, which behaves similarly to human logical thinking and provide an appealing interpretability. We take an example from OTC dataset, the feature *tra_name* has eight neighbor features, which means this feature has positive relevant with these eight features. In our real-life, the features *fund_code*, *profession_code* and *cust_code* effect the feature *trd_name*. *Currency_diversity* and *fund_intl* also effect the *tra_name*. This manner is consistent with human behavior.

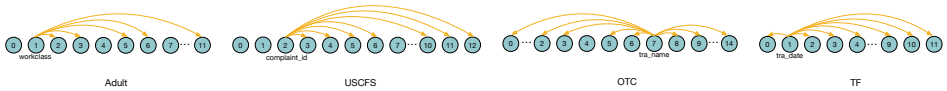


Fig. 11. Illustration of the feature similarity with an anchor feature in FM on four datasets. Each node has the same meaning as Fig.10. On four datasets, we compute the similarity of other features with an anchor feature. For example, on OTC dataset, we calculate the first eight features which have the greatest similarity with an anchor feature *tra_name*. These first eight features are *fund_code*, *channel_mode* and *class_mode* and so on. Obviously, there are three features *class_mode*, *fund_name* and *birthday* that are different from eight neighbor features generated by our GCFM. In real life, *class_mode* and *birthday* have little positive effect on the an anchor feature *tra_name*, especially the *birthday*. Moreover, in FM, the anchor feature has connection weights with all the remaining features except these first eight features, this behavior is prone to introduce noises.

(*i.e.*, Criteo, Avazu, USCFC and Adult) and two financial datasets that we collect offline in practical projects, (*i.e.*, OTC and TF). e extend our models to online practical real-world applications. Our GCFM method has been applied to a practical fund recommendation project by SenseTime Co., Ltd. Our project aims to offer accurate fund product recommendations to individual users.

Especially, there are two groups of customers. Each group has 1000 persons. These individuals are randomly selected from active users in realistic daily trading data for fund products during one year. We employ two strategies to recommend fund products for both groups. One strategy is *expert knowledge* and the other is our model GCMF. *Expert knowledge* denotes the suggestion come from financial human experts, who are professional in financial markets. Concretely, two fund managers, with financial backgrounds of master degree and the investment experience of 5 years. Both fund managers pick the top-5 fund products that the customers would be likely to buy according to users' purchase behaviors in recent a year. Our GCFM focuses on recommending their preferred fund products to users. In this paper, our GCFM is trained on the same historical daily trading data during one year; the trained GCFM excavates user preference to recommend the fund products to these customers.

Table 4 presents our experimental results. In the first group, our GCFM recommends 1000 persons for their interesting financial products, and 27 persons purchase financial products 92 times. Meanwhile, GCFM makes $14384.3 in profit. However, for fund managers of recommending financial products, there are just 2 persons buying products 1 time, which only makes $214.05 in profit. Obviously, in the second and third groups, GCFM is also more effective than expert knowledge.

TABLE 4
Results in real-world applications. We have three groups of experiments, and each group has 1000 persons. For each group, we recommend 1000 persons for their interested financial products with our GCFM and human experts.

Group	Retails (times)	Buyers (persons)	Revenue ($)
1-GCFM	92	27	14384.3
1-EXPERT	2	1	214.05
2-GCFM	76	24	12823.2
2-EXPERT	5	2	7463.2
3-GCFM	4	3	4444.6
3-EXPERT	0	0	0

6 Conclusion

This paper presents a novel Graph-Convolved Factorization Machine (GCFM) to improve RS performance via graph-based modeling of multi-feature interactions. Since conventional k-order feature interactions are extremely rigid, high-cost and easily leads to feature vanishing, to address these issues, the proposed GCFM models multi-feature interactions with a graph. Each feature needs to interact with its neighbor features and their connection weights can strengthen the closely related features and weaken the negatively irrelevant features. Furthermore, our GCFM achieves state-of-the-art performance and presents a superior interpretability in recommendation tasks. We further extend our model to online real-world applications, which show an appealing human-level decision intelligence in real scenarios. In the future, we will explore the possibility of adding causal reasoning to our GCFM to further improve personalized recommendation.

Acknowledgments

This work was supported in part by NSFC (No.62006253, U181146, 61836012), and State Key Development Program (No.2018YFC0830103).

References

[1] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: a survey," *Decision Support Systems*, vol. 74, pp. 12–32, 2015.

[2] H. Hwangbo, Y. S. Kim, and K. J. Cha, "Recommendation system development for fashion retail e-commerce," *Electronic Commerce Research and Applications*, vol. 28, pp. 94–101, 2018.

[3] Z. Zhao, H. Lu, D. Cai, X. He, and Y. Zhuang, "User preference learning for online social recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 9, pp. 2522–2534, 2016.

[4] G. Zhao, X. Qian, X. Lei, and T. Mei, "Service quality evaluation by exploring social users' contextual information," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 12, pp. 3382–3394, 2016.

[5] S. Rendle, "Factorization machines," in *International Conference on Data Mining*, 2010, pp. 995–1000.

[6] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for CTR prediction," in *International Joint Conference on Artificial Intelligence*, 2017, pp. 1725–1731.

[7] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," in *International Joint Conference on Artificial Intelligence*, 2017, pp. 3119–3125.

[8] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata, "Higher-order factorization machines," in *Neural Information Processing Systems*, 2016, pp. 3351–3359.

[9] Y. Juan, Y. Zhuang, W. Chin, and C. Lin, "Field-aware factorization machines for CTR prediction," in *ACM Conference on Recommender Systems*, 2016, pp. 43–50.

[10] X. He and T. Chua, "Neural factorization machines for sparse predictive analytics," in *ACM Research and Development in Information Retrieval*, 2017, pp. 355–364.

[11] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xdeepfm: Combining explicit and implicit feature interactions for recommender systems," in *ACM SIGKDD*, 2018.

[12] J. Pan, J. Xu, A. L. Ruiz, W. Zhao, S. Pan, Y. Sun, and Q. Lu, "Field-weighted factorization machines for click-through rate prediction in display advertising," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1349–1357.

[13] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 8609–8613.

[14] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, 2019.

[15] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon, "Collecting large, richly annotated facial-expression databases from movies," *IEEE MultiMedia*, vol. 19, no. 3, pp. 34–41, 2012.

[16] H. Kim, G. Y. Kim, and J. Y. Kim, "Music recommendation system using human activity recognition from accelerometer data," *IEEE Trans. Consumer Electronics*, vol. 65, no. 3, pp. 349–358, 2019.

[17] L. Luo, H. Xie, Y. Rao, and F. L. Wang, "Personalized recommendation by matrix co-factorization with tags and time information," *Expert Syst. Appl.*, vol. 119, pp. 311–321, 2019.

[18] L. Zervakis, C. Tryfonopoulos, S. Skiadopoulos, and M. Koubarakis, "Query reorganization algorithms for efficient boolean information filtering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 2, pp. 418–432, 2017.

[19] B. Hong and M. Yu, "A collaborative filtering algorithm based on correlation coefficient," *Neural Computing and Applications*, vol. 31, no. 12, pp. 8317–8326, 2019.

[20] D. Lian, Y. Ge, F. Zhang, N. J. Yuan, X. Xie, T. Zhou, and Y. Rui, "Scalable content-aware collaborative filtering for location recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 6, pp. 1122–1135, 2018.

[21] Q. Liu, S. Wu, and L. Wang, "Multi-behavioral sequential prediction with recurrent log-bilinear model," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 6, pp. 1254–1267, 2017.

[22] J. Chen, C. Wang, J. Wang, and P. S. Yu, "Recommendation for repeat consumption from user implicit feedback," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 11, pp. 3083–3097, 2016.

[23] P. Ashokkumar, N. Arunkumar, and S. Don, "Intelligent optimal route recommendation among heterogeneous objects with keywords," *Computers & Electrical Engineering*, vol. 68, pp. 526–535, 2018.

[24] L. Gao, J. Wu, C. Zhou, and Y. Hu, "Collaborative dynamic sparse topic regression with user profile evolution for item recommendation," in *AAAI*, 2017.

[25] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys*, vol. 52, no. 1, p. 5, 2019.

[26] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *ACM International Conference on Multimedia*, 2014, pp. 627–636.

[27] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," in *Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5308–5317.

[28] S. Li, Z. Yan, X. Wu, A. Li, and B. Zhou, "A method of emotional analysis of movie based on convolution neural network and bi-directional LSTM RNN," in *IEEE International Conference on Data Science in Cyberspace*, 2017, pp. 156–161.

[29] Q. Li, X. Yao, and C. Wang, "RBMP: a relay-based MAC protocol for nanonetworks in the terahertz band," in *ACM International Conference on Nanoscale Computing and Communication*, 2017.

[30] H. Ye, X. Zheng, and C. Rong, "Hybridization of pmf and lstm for recommendation of intelligent resource," in *International Conference on Green Informatics*, 2017, pp. 6–10.

[31] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *ACM International Conference on Multimedia*, 2014, pp. 627–636.

[32] M. Pichl and E. Zangerle, "Latent feature combination for multi-context music recommendation," in *IEEE International Conference on Content-Based Multimedia Indexing*, 2018, pp. 1–6.

[33] H. T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide&deep learning for recommender systems," in *ACM deep learning for recommender systems*, 2016, pp. 7–10.

[34] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *ACM ADKDD*, 2017.

[35] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1059–1068.

[36] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang, "Autoint: Automatic feature interaction learning via self-attentive neural networks," in *ACM International Conference on Information and Knowledge Management*, 2019, pp. 1161–1170.

[37] T. Huang, Z. Zhang, and J. Zhang, "Fibinet: combining feature importance and bilinear feature interaction for click-through rate prediction," in *ACM Conference on Recommender Systems*, 2019, pp. 169–177.

[38] M. Blondel, M. Ishihata, A. Fujino, and N. Ueda, "Polynomial networks and factorization machines: New insights and efficient training algorithms," in *International Conference on Machine Learning*, 2016, pp. 850–858.

[39] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web International Conference*, 2018, pp. 593–607.

[40] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Neural Information Processing Systems*, 2016.

[41] S. Ryu, Y. Kwon, and W. Y. Kim, "A bayesian graph convolutional network for reliable prediction of molecular properties with uncertainty quantification," *Chemical Science*, vol. 10, 2019.

[42] L. Wu, P. Sun, R. Hong, Y. Fu, X. Wang, and M. Wang, "Socialgcn: An efficient graph convolutional network based model for social recommendation," *the Computing Research Repository*, vol. abs/1811.02815, 2018.

[43] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 974–983.

[44] C. Jeong, S. Jang, H. Shin, E. Park, and S. Choi, "A context-aware citation recommendation model with BERT and graph convolutional networks," *the Computing Research Repository*, vol. abs/1903.06464, 2019.

[45] H. Wang, D. Lian, and Y. Ge, "Binarized collaborative filtering with distilling graph convolutional networks," *the Computing Research Repository*, vol. abs/1906.01829, 2019.

[46] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[47] F. Li, Z. Chen, P. Wang, Y. Ren, D. Zhang, and X. Zhu, "Graph intention network for click-through rate prediction in sponsored search," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 961–964.

[48] J. Sun, Y. Zhang, C. Ma, M. Coates, H. Guo, R. Tang, and X. He, "Multi-graph convolution collaborative filtering," in *IEEE International Conference on Data Mining (ICDM)*, 2019.

[49] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *International ACM SIGIR conference on Research and development in Information Retrieval*, 2019.

[50] Z. Li, Z. Cui, S. Wu, X. Zhang, and L. Wang, "Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction," in *ACM International Conference on Information and Knowledge Management*, 2019, pp. 539–548.

[51] F. Braida, C. E. Mello, M. B. Pasinato, and G. Zimbrão, "Transforming collaborative filtering into supervised learning," *Expert Systems with Applications*, vol. 42, no. 10, pp. 4733–4742, 2015.

[52] E. Greenshtein, Y. Ritov *et al.*, "Persistence in high-dimensional linear predictor selection and the virtue of overparametrization," *Bernoulli*, vol. 10, no. 6, pp. 971–988, 2004.

[53] R. A. Harshman *et al.*, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis," 1970.

[54] K. L. Elmore and M. B. Richman, "Euclidean distance as a similarity metric for principal component analysis," *Monthly Weather Review*, vol. 129, no. 3, pp. 540–549, 2010.

[55] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors A multilevel approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1944–1957, 2007.

[56] K. Kawaguchi, L. P. Kaelbling, and T. Lozano-Pérez, "Bayesian optimization with exponential convergence," *the Computing Research Repository*, vol. abs/1604.01348, 2016.

[57] S. P. Siregar and A. Wanto, "Analysis of artificial neural network accuracy using backpropagation algorithm in predicting process (forecasting)," *International Journal Of Information System & Technology*, vol. 1, no. 1, pp. 34–42, 2017.

[58] E. W. Steyerberg, "Overfitting and optimism in prediction models," in *Clinical prediction models*, 2019, pp. 95–112.

[59] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

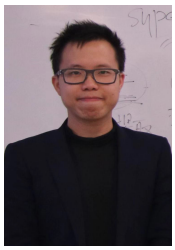
[60] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Symposium on Operating Systems Design and Implementation*, 2016, pp. 265–283.

[61] G. Nahler, "two-tailed test," *Springer Vienna*, 2009.

[62] J. L. B. Diederik P. Kingma, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

Yongsen Zheng received her M.S. degree in software engineering from Sun Yat-sen University, Guangzhou, China, in 2016. She is currently pursuing a Ph.D. degree in computer science and technology at Sun Yat-sen University, advised by Professor Liang Lin. Her current research interests include recommendation systems, knowledge discovery, behavior analysis and machine learning.

Pengxu Wei received the B.S. degree in computer science and technology from the China University of Mining and Technology, Beijing, China, in 2011 and the Ph.D. degree from University of Chinese Academy of Sciences in 2018. Her current research interests include computer vision and machine learning. She is currently a research scientist at Sun Yat-sen University.



Ziliang Chen received his B.S. degree in mathematics and applied mathematics from Sun Yat-sen University, Guangzhou, China. He is currently pursuing a Ph.D. degree in computer science and technology at Sun Yat-sen University, advised by Professor Liang Lin. His current research interests include computer vision and machine learning.

Yang Cao received her Ph.D. in theoretical physics from the National University of Singapore (NUS) and a B.Sc. from the University of Science and Technology of China (USTC). She is currently a research director at SenseTime, where she leads the AI in Finance team. Her research interests include recommendation systems, graph embedding, knowledge discovery and behavior analysis. Her Ph.D. work was on higher-order derivatives for option pricing. Before joining SenseTime, she was the Senior Vice President and Principal Data Scientist at GIC (Government of Singapore Investment Corporation, a sovereign wealth fund) and lead of the Econs Tech Family at Grab (a transportation unicorn company in Southeast Asia).

Liang Lin is a Full Professor of computer science at Sun Yat-sen University. He served as the Executive Director and Distinguished Scientist of SenseTime Group from 2016 to 2018, leading the R&D teams for cutting-edge technology transferring. He has authored or co-authored more than 200 papers in leading academic journals and conferences, and his papers have been cited by more than 16,000 times. He is an associate editor of IEEE Trans. Neural Networks and Learning Systems and IEEE Trans. Human-Machine Systems, and served as Area Chairs for numerous conferences such as CVPR, ICCV, SIGKDD and AAAI. He is the recipient of numerous awards and honors including Wu Wen-Jun Artificial Intelligence Award, the First Prize of China Society of Image and Graphics, ICCV Best Paper Nomination in 2019, Annual Best Paper Award by Pattern Recognition (Elsevier) in 2018, Best Paper Dimond Award in IEEE ICME 2017, Google Faculty Award in 2012. His supervised PhD students received ACM China Doctoral Dissertation Award, CCF Best Doctoral Dissertation and CAAI Best Doctoral Dissertation. He is a Fellow of IET.